Lab 3: Recursive Descent Parser

These solutions available in an html version[1] or a pdf version[2].

Adding productions to the grammar

- What production should be added to handle the sentence "He ate salad" ?

  Add the production `Pro -> 'He'`

- Is there a problem with either of the parse trees ?

  The parse tree for "he ate salad with a fork" is correct where the preposition phrase "with a fork" is correctly attached to the verb ('high' attachment). For the sentence "he ate salad with mushrooms", the preposition phrase "with mushrooms" is wrongly attached to the verb, whereas it should be attached to the noun phrase "salad" ('low' attachment).

- Change the order of the rules "NP -> N" and "NP -> NP PP"

  This leads into a classic problem of left recursion in depth-first search, where the `NP -> NP PP` production is applied infinitely. When this rule is ordered after the other possibilities, a parse is found before the left recursion problem occurs. However, if you try to get the parser to find further parses (or if you run it on a sentence for which no parse is possible), it will eventually hit the same problem of infinite recursion.

Ungrammatical sentences

- Though the second sentence is ungrammatical, it parsed by our grammar. Modify the grammar to handle such cases

  Change `VP -> V | V NP | V NP PP` to `VP -> Vi | Vt NP | Vp NP PP`. This will handle the subcategorization information of verbs correctly.

Number agreement (optional)

See our `grammar2` in the answer code[3].

Extracting and viewing parsed sentences

- What is the type of the parsed sentence object?

  `type(psents[0])` gives the object type which is `nltk.tree.Tree`

- Extract the list of words and the list of word,pos-tag tuples from `psents[0]` using some of the other available methods.

  `psents[0].leaves()` and *psents[0].pos()*' will gives the list of words and word, pos-tag tuples

[1] http://homepages.inf.ed.ac.uk/sgwater/teaching/lsa2015/labs/lab3-sol.html

[2] http://homepages.inf.ed.ac.uk/sgwater/teaching/lsa2015/labs/lab3-sol.pdf

[3] lab3-sol.py

Distribution of Productions

See our answer code[3] for the `production_distribution` function.

- Do you expect there to be more lexical or non-lexical (grammatical) productions in the grammar? How many productions of each type are there actually?

  We would probably expect more lexical productions, since there is at least one lexical production for every word in the vocabulary. However you might be surprised at how many grammatical productions there are (7982, vs 13781 lexical productions): treebank grammars are not small!

- What are the 10 most frequent and least frequent lexical and grammatical productions? Plot histograms of the top 50-100 productions of each type. Do these distributions look familiar?

  Again, see the answer code. As usual, the distributions appear to be roughly Zipfian (which is obvious for the lexical productions, since they are in fact words, but might not have been so obvious to you for the non-lexical productions).